

Getting Clear About Fuzzy Matching

A GUIDE TO MATCHING ADMINISTRATIVE DATA SETS

March 2022

CONTACT

Taylor Doren, MA
Taylor.Doren@providence.org

CORE TEAM

Taylor Doren, MA
Chia-Hua Yu, MBI
Ben Gronowski, MHR

Contents

What Is Fuzzy Matching & Why Does it Matter?.....	3
What to Expect from This Guide.....	4
Fuzzy Matching in Greater Detail.....	5
The Steps to Fuzzy Matching.....	6
Step 1. Profile Your Data.....	8
Step 2. Standardize Your Files.....	10
Step 3. Cleanse Your Data.....	11
Step 4. Create Composite Keys.....	13
Step 5. Fuzzy Match.....	14
Step 6. Manually Review.....	21
Step 7. Iterate.....	25
Case Study: Lessons Learned on a Fuzzy Matching Project.....	26
Frequently Asked Questions.....	28
Appendix A. Additional Suggestions When Cleansing Your Data.....	34
Appendix B. An Example Workflow: What This Process Looks Like at CORE.....	36

Fuzzy Matching & Why It Matters

As anyone with experience working with administrative data sets can attest, administrative data can be dirty, incomplete, and change over time. Even within a single data set, individuals can have multiple records, each containing slightly different information. When working across multiple data sets, these problems only multiply. Individuals' names may change over time or be spelled differently across data sets, addresses may be formatted in different ways, or birth dates might be inconsistent, to name only a few challenges.

Fuzzy matching allows records across data sets and sources to be linked, even when the original data are less than perfect. By combining deterministic and probabilistic methods, fuzzy matching allows you to establish whether multiple entries within or across data sets refer to the same entity. Fuzzy matching can be used to identify people within a household or be used to identify individuals. Fuzzy matching can be leveraged to link data within a single administrative data source, but it is particularly powerful when working with cross-sector data to assign unique identifiers to individuals appearing across data sets and systems. By combining, standardizing, blending, and matching individual participant rows from across systems, fuzzy matching allows you to establish a unique identifier, which can then be attached to all rows associated with an individual or household across systems.

While fuzzy matching techniques are most often applied for marketing and sales purposes, we've leveraged the concept to compare individuals' records from disparate data sources and answer questions for cross-sector data projects related to social determinants of health research. Cross-sector research projects with fuzzy matching methods applied allow a shared unique identifier to be assigned for all the individuals present in each of the disparate data sets. Once unique identifiers are created for individuals across all the data, you can start to answer questions like:

- ▶ Which door do most folks walk through to access support?
- ▶ How does getting support in one system translate to utilization in other systems?
- ▶ How can programs strategically align their efforts to best support the clients that they share?

Without fuzzy matching, it would be incredibly challenging to answer these questions (and many more) due to inconsistent/incomplete administrative data housed in data silos.

Fuzzy matching allows you to link individuals across disparate data sources to explore research questions, which can result in more wholistic understandings of the communities that programs serve, more effective outreach efforts, increased communication across systems, improved individual and population outcomes, and cost savings.

What to Expect from This Guide

In this guide, we draw on the knowledge that we've gained at CORE working within and across administrative data sets to identify a new unique identifier for individuals on a wide variety of projects. These include matching using housing, Medicaid, school district, and sheriff's office booking data; range from small projects with a single data set to projects spanning multiple data sets where we've matched over 850,000 rows; as well as one-time projects with static data and projects with dynamic data and periodic data refreshes. Regardless of the content, size, and frequency of the data sources being matched, fuzzy matching has been invaluable in helping us answer big questions about how individuals use systems and how these systems can better understand and serve their client populations.

While fuzzy matching can be done on all types of data sets, in this guide we focus on matching administrative data, given the nature of our expertise and work. Administrative data are inherently dirty, so we focus heavily on the importance and process of standardizing and cleansing your data. Administrative data typically includes both elements describing a person (e.g., name, DOB, and SSN) and where they live (e.g., residential address), and we cover both types of these data in this guide. Additionally, different tools can be used in the fuzzy matching process. While SQL and Alteryx are the two tools that we focus on here, we believe that the approach and considerations outlined herein are valuable even if you have access to different resources. If you work in a smaller organization and your data all live in a spreadsheet, be sure to check out CORE's slide deck walkthrough of steps to generate a unique identifier for individuals within a spreadsheet!

There are many areas that require special attention when working with administrative data, which include types of data, data entry requirements, your project's goals, organizational resources and limitations, and the populations represented in the data that you're working with. In this guide, we provide a general overview of the fuzzy matching process and give examples of frequently seen challenges related to these areas, although we recognize that we cannot possibly cover all the unique situations you might face. By the end of this document, we hope that you feel like you have the knowledge needed to apply this process, or elements of this process, in your own work.

About CORE

The Center for Outcomes Research and Education (CORE) is an independent team of scientists, researchers, and data experts with a vision for a healthier, more equitable future. We work with changemakers and communities to take on today's biggest barriers to better health – from the COVID-19 pandemic to the housing crisis and more.

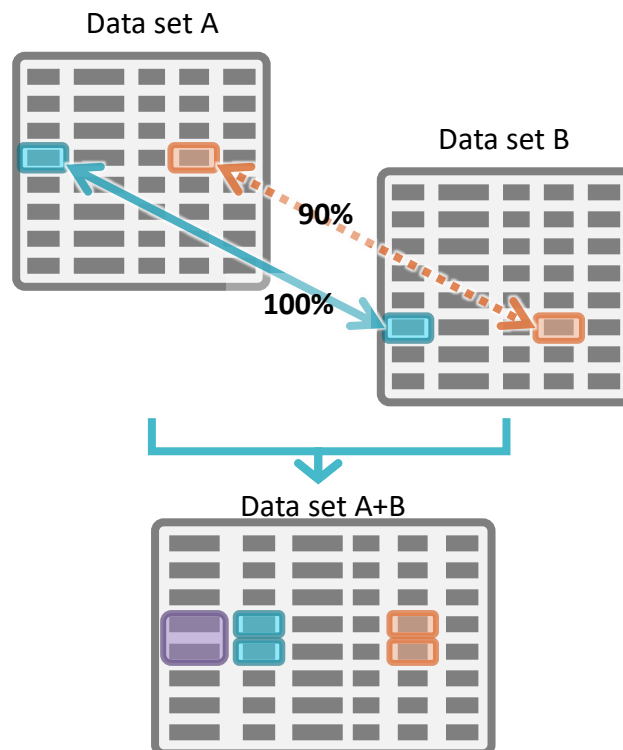
At CORE, we know that the opportunity for health begins long before illness. That's why so much of our research focuses on the conditions in which people live, work, learn, and play. These conditions, known as the Social Determinants of Health, are a major driver of health outcomes and health disparities. We seek to understand and address these conditions and their driving factors, so we can build evidence that leads to meaningful improvements in health and equity.

Fuzzy Matching in Greater Detail

Fuzzy matching combines deterministic and probabilistic methods to try to establish whether two or more entries within or across data sets refer to the same individual.

Deterministic methods rely on exact matches between data sources for key identifiers, such as a person's name, date of birth (DOB), social security number (SSN), or other unique identifier in the system. When matching exact values like these, it can be said that an entity is *determined* to be the same across data systems.

Probabilistic methods, while slightly less definitive than deterministic methods, allow for matching when the identifiers being matched are less reliable. This might include inconsistent names or DOBs across systems. In these cases, it can be said that there is a high degree of *probability* that the individual is the same across data systems.



By leveraging these two methods, fuzzy matching can help you determine whether person 1 in data set A is the same as person 2 in data set B. Assuming they are, you can create a shared data set and apply a new unique identity to this individual, transforming person 1 and person 2 into **person A**.

The same principles apply when matching non-person data, although each type of data has its own unique considerations. We outline those considerations in the Frequently Asked Questions section of this guide.

The Steps to Fuzzy Matching

The fuzzy matching process involves pre-matching prep work, creating a model with specific fuzzy matching rules and algorithms, and then manually reviewing results. While we lay out the process as a series of ordered actions, fuzzy matching involves more than just checking off each step. A key part of the process is iterating on the models you develop and repeating the matching and manual review steps until you're satisfied with the results. We refer to the iterative process as Step 7 in this document for simplicity's sake, while recognizing that it will take place more than once, both before and after other steps.

Pre-Matching Prep Work

- ▶ **Step 1 – Profile Your Data:** First, familiarize yourself with the data that will be combined. Familiarity with available fields that you could match on, the presence of existing unique individual identifiers in the data, overall cleanliness of the source data, and expected rates of matching across data sources are great goals for this step.
- ▶ **Step 2: Standardize Your Files:** Next, configure the data you want to match so that it's formatted similarly across sources. As part of this process, you should implement a process to track changes made back to the original data. Here your objective is to have the same number of columns, column names, and field types across your various data sources.
- ▶ **Step 3 – Cleanse Your Data:** In step 3, you develop data cleansing rules and apply those rules to all source data. Doing this after step 2 ensures that the data are uniform and increases the probability that values match across sources. At this stage, developing and applying rules based on the unique data that you're working with are your goals.
- ▶ **Step 4 – Create Composite Keys:** By combining values from different standardized and cleansed fields, you create a single string in this step. While we find this step valuable as it increases the likelihood of finding matches, it is dependent on your access to specific resources. As such, it the only pre-matching step that we consider optional. Here, your goal is to create multiple unique composite keys by blending different fields.

Fuzzy Matching

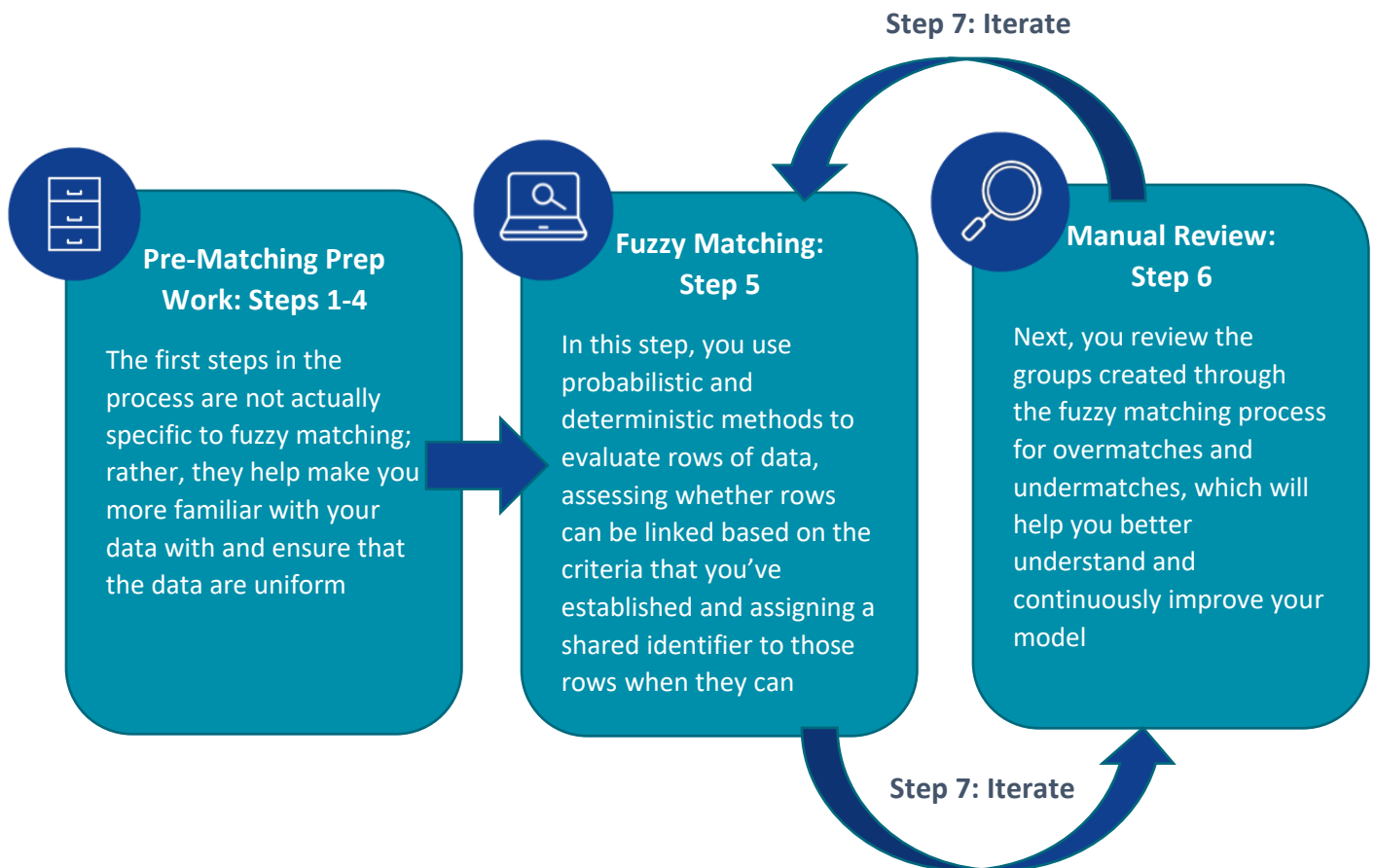
- ▶ **Step 5 – Fuzzy Match:** In this step you put all your pre-matching prep work into practice – by using probabilistic and deterministic methods to evaluate rows of data, you decide whether rows can be linked based on the criteria that you established. Your objective here is to determine matching criteria (create a model) and use those criteria to match across all rows in your data set. By the end of the fuzzy matching process, you should have a new identifier that's shared among all the rows that were probabilistically and deterministically linked.

Manual Review

- **Step 6 – Manually Review:** After completing the initial round of fuzzy matching, you review your results manually, providing insight into how your model performed and ways in which it could be improved. Understanding the extent to which overmatching (multiple records erroneously matched) and undermatching (multiple records erroneously unmatched) occurred is an excellent goal for this step.

Iterate

- **Step 7 – Iterate:** Using what you learned from the manual review process in Step 6, you can iterate with your model by selecting different matching rules until you're ready to run the model (or run the model again) on your full data set. Your goal in this step is to take your new knowledge and go back to Step 5 (fuzzy matching) and Step 6 (manual review), improving your model each time.



Step 1: Profile Your Data

To make the matching process smoother and more efficient, the first step is to familiarize yourself with the data that you'll be combining. Each data source will have idiosyncrasies inherent in the data, so getting to know what those are will be important. Additionally, most source data will be populated with a person-level identifier. This identifier will be key during the matching process and will be referred to within this guide as the SourceID. The following questions might be helpful to consider during this data profiling process, although they are by no means comprehensive.

What fields do you have?

- ▶ Do you have demographic information (such as name, DOB, SSN) and addresses, or just demographic information?
- ▶ If working with multiple source files, do the files have the same demographic fields?

How complete are the fields that you have?

- ▶ If you have social security numbers, do you have all 9 characters or just the last 4?
- ▶ If you have names, do you have first, middle, and last names, or just first or middle initials, for example?

How are those fields presented?

- ▶ Is demographic information in a single field or parsed into separate fields?
 - ❖ *Example:* Name in a single field (Name): "John J Smith" versus separate name fields (FirstName, MiddleName, LastName): "John", "J", "Smith".

What are the most frequently populated values in each field?

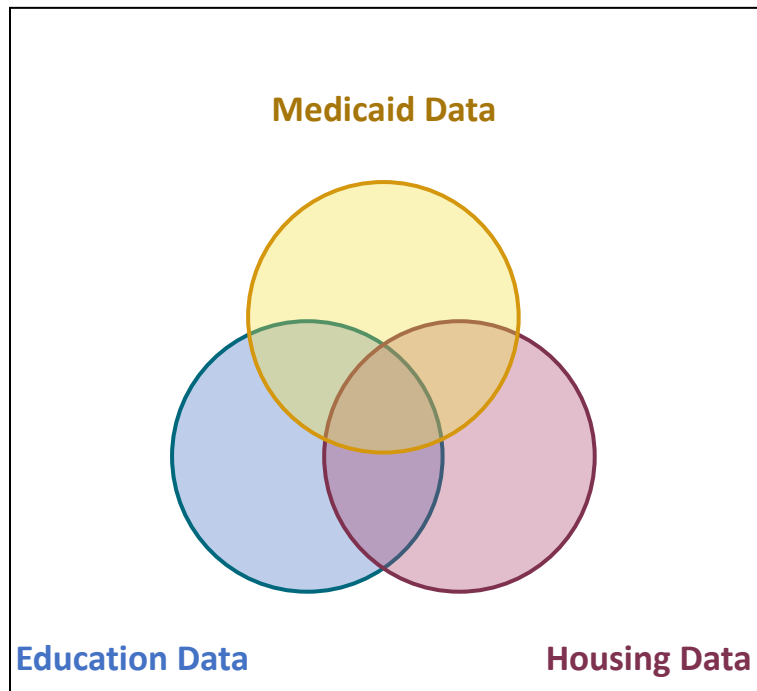
- ▶ Are there 'dummy values' present?
 - ❖ *Example:* Dummy values are values used during data entry to move past required fields, which are not actually unique identifiers for the individual represented. "123456789" as an SSN or "1/1/1911" as a DOB are examples of dummy values. We address what to do with dummy values during data cleansing (Step 3).

Is the person-level ID that's present in the source data actually unique?

- ▶ If there are multiple rows per person in your source file, does each row have its own SourceID or is the SourceID shared across all the rows for the same person?
 - ❖ *Example:* John Jacob Smith with a DOB of 1/1/1950 has 3 rows of source data. Either the 3 rows all have the same SourceID (which is preferred) or they are not consistent with each other (which is not preferred).

What types of populations are in your files, and how likely are they to match across data sets?

- ▶ Do you expect only a subset of people from one data set to match another?
 - ❖ *Example:* Matching education data, which just captures children, to housing data, which may include children, but they won't be named as the primary individual.
- ▶ Is it more likely that there will be a lot of matches across the data sets?
 - ❖ *Example:* Matching housing data, which may include either the head of household or the full family, to Medicaid data, which is more likely to include the full family.



Step 2: Standardize Your Files

Standardizing your files ensures consistent formatting across data sources. This includes standardizing column names and field types across source files. You might also have to create columns to make your data sets consistent if certain columns are present in one source file and not another. We have found it easiest to combine all rows from all files into a single table used within the fuzzy matching model, even if some of the sources have blank or null values for any of the given fields. Combining all the rows of data into a single input for your fuzzy matching solution and creating a single auto-incrementing primary key for the table (referred to as RecordID in this guide) value cuts down on potential errors. When you combine your different data tables, make sure you add a field to indicate where the row originally came from. This includes pulling the unique ID from the source data (SourceID) and creating a column that tracks where each row was sourced from, named something like DataSource. These two identifiers (the source ID and the source name) should be enough to get you back to the respective source table. If they're not, add additional breadcrumbs to help you get there.

We have found it easiest to combine all rows from all files into a single table used within the fuzzy matching model

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	RecordID	SourceID	DataSrc	FirstName	LastName	MiddleName	Sex	DOB	SSN	Addr	City	State	Zip

This visual shows the fields from the single combined table. The columns on the far left in the darker color are the minimum starting set you should use to fuzzy match. Ideally, you'll have some additional columns! Pairing the leftmost columns with some from the middle columns and/or rightmost columns (lighter colors) will be best in the absence of having access to all the fields shown.

Step 3: Cleanse Your Data

When profiling your data (Step 1), you might have discovered that your source data aren't as clean as you'd like. Fields that you'd like to be combined might be separate, fields that you'd like to be separate might be combined, or data might be formatted in ways that make it hard to work with. This is especially true when matching data from different sources, where each source might have its own unique standards that don't quite match those of other sources.

In this step you create data cleansing rules and apply those rules across your data source, which ensures that your data are standardized. Be sure to keep your original data and add fields to your table that contain the cleansed values based on these suggestions. This will save you time, energy, and resources when you're ready to fuzzy match. We list a few of the most common cases and suggested solutions below, but additional challenges and recommendations can be found in Appendix A.

Upper case all values

- ▶ *Original:* Smith
- ▶ *Cleansed:* SMITH
- ▶ *Rationale:* Some administrative data might be in all caps originally and some might not. Applying a standard rule to ensure that the letter case is the same across data sources can be helpful when matching.

Remove all leading and trailing white spaces

- ▶ *Example:* __SMITH__
- ▶ *Cleansed:* SMITH
- ▶ *Rationale:* Some tools might not consider a match if there are extra spaces in the string. This can be true even if the characters are an exact match, with the only difference between strings being an additional space.

Split hyphenated last names into separate fields

- ▶ *Example:* JONES-SMITH
- ▶ *Cleansed:* FirstName: JONES, SecondLastName: SMITH, CleanLastName: JONESSMITH
- ▶ *Rationale:* Some last names change over time based on life experiences such as marriage or divorce, comparing the last name segments separately can help with matching.

Remove accent marks and punctuation from names columns

- ▶ *Example:* JOSÉ
- ▶ *Cleansed:* FirstName: JOSE

- ▶ *Rationale:* Some systems do not allow for accent marks or other punctuation marks, or they're entered in the data systems inconsistently. Removing accent marks and punctuation is one way to standardize how the names will be treated in the matching algorithms.

Parse addresses into separate columns

- ▶ *Example:* 123 POPLAR STREET, PORTLAND, OR 97212
- ▶ *Cleansed:* StreetAddress1: 123 POPLAR STREET, City: PORTLAND, State: OREGON, ZIP: 97212
- ▶ *Rationale:* Like parsing names into separate columns, ideally the address data that you receive will be parsed when you receive it. If it is not, creating separate columns for street address, city, state, and zip code will be an important step in formatting your data.

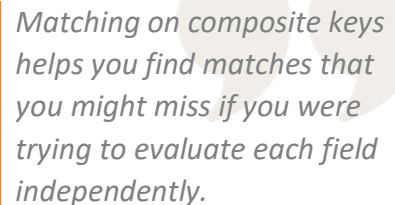
Make dummy values blank or null

- ▶ *Example:* 123456789 as an SSN
- ▶ *Cleansed:* _____ as an SSN
- ▶ *Rationale:* Fields with dummy values won't actually help you match values since they're not real data. If dummy values are retained, you can end up matching records erroneously because the dummy values are likely shared across multiple individuals' records. By making these fields blank or null, you can then focus on matching the substantive data that you do have.

Step 4: Create Composite Keys

After standardizing your data into a single table (step 2) and cleansing your data (step 3), you can combine fields to create composite matching keys. These keys take values from various fields and combine them into a single string. In this step, you combine different fields in different orders, which will give you many distinct composite keys. Matching on composite keys helps you find matches that you might miss if you were trying to evaluate each field independently. For example, sometimes names are entered into systems with the last name populated in the first name field and vice versa. When a composite key is created, it becomes a single field to be compared in probabilistic fuzzy matching algorithms, rather than discrete fields that cannot be compared. By including composite keys in your model, fuzzy matching algorithms may be able to identify a row with a transposed name as a match to a different row that does not have the transposed name.

This process can be done with whatever tool you have at your disposal. While this step is considered optional and requires access to these resources, creating composite keys is highly encouraged since it increases your likelihood of identifying matches. Below are a few examples of composite matching keys to give you an idea of what this looks like, but all sorts of different composite keys can and should be created for potential matching.



Matching on composite keys helps you find matches that you might miss if you were trying to evaluate each field independently.

Combine name, DOB, and SSN

- ▶ *Combination:* LastNameClean + FirstNameClean + DOBDigits + SSNClean
- ▶ *Example:* SMITHJOHN20000101123456789

Combine DOB, SSN, sex, and partial name

- ▶ *Combination:* DOBDigits + SSNClean + Sex + LastNameClean + FirstName3
- ▶ *Example:* 20000101123456789MSMITHJOH

Combine name, partial address, partial DOB, and partial SSN

- ▶ *Combination:* LastNameClean + FirstNameClean + AddrNumbClean + YearDOB + SSN4
- ▶ *Example:* SMITHJOHN12320006789

Step 5: Fuzzy Match

This is the step that you've likely been waiting for – doing the fuzzy matching! In this step, you use probabilistic and deterministic methods to evaluate rows of data and determine whether they can be linked based on the criteria that you established. If they can be linked, you create and attach a new identifier that's shared across those rows. Try different combinations of data to fuzzy match; record linkage during fuzzy matching is informed in part by the way you sort your data and which data elements you're comparing. Fuzzy matching is not a linear process, and there shouldn't be a single way you're evaluating your data. Make sure you compare your data with different combinations of data elements and combine all the results from the various reviews into your newly created ID.

What this looks like in practice will depend on the data that you're working with and the tool that you're using; there are some tools that have built-in fuzzy matching capabilities such as RedPoint, Informatica, or Alteryx. These specialized tools can run different algorithms in parallel to output MatchedID values, whereas if you're working in SQL, you're more likely to run your algorithms in sequential scripts. For more benefits and downsides of SQL and the specialized tools, see the box to the right.

In either SQL or these specialized tools, you can leverage the cleansed fields you developed (Step 3) and the composite fields you created (Step 4) to match with different common fuzzy matching algorithms. These algorithms include methods to encode or otherwise standardize string values (such as Soundex, NYSIIS, Metaphone, and Double Metaphone), which make it easier to then compare generated values. Algorithms based on pattern matching (such as Jaro distance or Levenshtein distance) can also be helpful. These algorithms measure the edit distance between strings (Levenshtein distance), or perform calculations based on common characters and the number of transpositions (Jaro distance). There are several readily available resources on the Internet that can further explain these methods, as well as the code needed to create SQL functions for non-native SQL algorithms (like Jaro or Levenshtein).

SQL versus Specialized Fuzzy Matching Tools

When deciding what tool best meets your needs, considerations will likely include cost, accessibility, and the learning curve associated with the tool.

SQL's upsides are that it's low-cost, likely a tool that you're already using and, therefore, more accessible, and straightforward. Its downsides are the time and resources required when matching on large amounts of data.

Specialized tools are highly customizable and easier to use for repeated matches. You can create a suite of fuzzy matching building blocks that can be substituted in or out of your model, based on your needs. The downside of these tools is their cost, which can be prohibitive for some organizations. Additionally, there can be more of a learning curve when starting out with these types of specialized tools.

Regardless of whether you're matching using SQL or a specialized tool, you can still fuzzy match successfully. Below we describe steps to take irrespective of the tool that you're using, and then we dive into the process for each tool:

Establish a use case and subsample

- *Use cases:* To start, you want to identify a few individuals as use cases and follow them through your fuzzy matching process. Ideally, these individuals have complex or varied fields in the data set. The rationale behind this is if you can match successfully for complex cases, you can have greater certainty in other matches. It will also save you some time as you build out your SQL code or workflow to work with a few individuals rather than the entire data set.

Determine whether your source has a unique identifier

- *SourceIDs:* If any of your original data sources list the same individual with multiple SourceIDs, it is best to first fuzzy match within the source file(s) to generate a MatchedID. This MatchedID can then be set as the SourceID in the larger data table to be fuzzy matched.

If matching using SQL:

- ▶ **Formatting:** You might not use composite keys if you're matching in SQL, but you should make sure that you're using cleansed fields. If you did not standardize your source data in step 2, do so now: The first step is to combine all rows into a single table and create a column titled RecordID, which will be the primary key (a unique identifier for each row) for your table. You then join this table to itself with each set of criteria that you use; you can start with the strictest criteria and then loosen them with each iteration.
- ▶ **Initial matching:** Use the strictest matching criteria available to start to give you a high degree of confidence in your matches. For example, if you match on exact first name, exact last name, exact DOB, and exact SSN, you can feel certain that any matches returned are the same person. On the other hand, if you match on the first three letters of the first name, the first four letters of the last name, DOB year, and zip code, you might not feel quite as confident about the matches returned.

❖ **Example:** Exact match on LastNameClean + FirstNameClean + SSN + DOBDigits:

```
from matching.DataTable a join matching.DataTable b
on b.LN = a.LN and b.FN = a.FN and b.DOB = a.DOB and b.SSN =
a.SSN and b.RecordID <> a.RecordID
```

- ▶ **Tracking matches:** Next, create a MatchedID column to tag and track your matches. Our practice is to use the smallest RecordID value of the matched group as the MatchedID. This way, the MatchedID will be consistent for each row in the group that is matched. You want to include your matched rows in future matching iterations, since there might be information on a previously matched row that's helpful in matching a currently unmatched row.

❖ **Example:**

```
select distinct min(a.RecordID)
over(partition by a.LN, a.FN,
a.DOB, a.SSN order by a.LN, a.FN,
a.DOB, a.SSN) MatchedID, a.LN,
a.FN, a.DOB, a.SSN, a.RecordID,
b.RecordID
into matching.DataTableMatched
from matching.DataTable a
join matching.DataTable b
```

You want to identify a few individuals as use cases and follow them through your fuzzy matching process. Ideally, these individuals have complex or varied fields in the data set.

Match ID	Record ID
0001	0001
0002	0002
0004	0004
0004	0006
0004	0009
0011	0011
0012	0012
0012	0013
0014	0014
0014	0019


```
on b.LN = a.LN and b.FN = a.FN and b.DOB = a.DOB and b.SSN =
a.SSN and b.RecordID <> a.RecordID
```

- *Expand your matching criteria:* Next, expand your matching criteria slightly and include your newly created MatchedID. Be sure to keep all your rows, even if they matched in the previous step(s). There might be elements present on the matched rows that help match to other rows. You can also test your results for values that are similar but not exact matches through phonetic matching algorithms (like Soundex, NYSIIS, Metaphone, and Double Metaphone), which will let you match on values that are different but within specified criteria. You can update the table with any newly found MatchedIDs based on the looser criteria.

- ❖ *Example:* Match on LastNameClean + DOBDigits + DIFFERENCE(FirstNameClean) >=3:

```
from matching.DataTableMatched a
join matching.DataTableMatched b
on b.LN = a.LN and b.DOB = a.DOB and difference(a.FN, b.FN) >= 3
and b.RecordID <> a.RecordID and b.MatchedID <> a.MatchedID
```

- *Continue expanding your matching criteria:* Eventually you will find the loosest possible criteria that you feel comfortable with. The looser the criteria, the higher chance you have of overmatching (matching two different people as the same individual). As you loosen your criteria, manual review becomes even more important. Keep updating the MatchedID column with the smallest RecordIDs of the group.

- ❖ *Example:* Match on LastName4 + FirstName3 + MonthDOB + DayDOB:

```
from matching.DataTableMatched a
join matching.DataTableMatched b
on b.LN4 = a.LN4 and b.FN3 = b.FN3
and left(convert(varchar, b.DOB, 110), 5) =
left(convert(varchar, a.DOB, 110), 5)
and b.RecordID <> a.RecordID and b.MatchedID <> a.MatchedID
```

- *Consider using Levenshtein or Jaro functions:* While the previous examples may seem rather deterministic, there is some degree of probability when comparing parts of DOBs or the beginning of first or last names. You can also find code online to create SQL functions that use more traditional probabilistic algorithms. You should follow a similar process as described above where you sequentially start with the strongest rules first and apply looser criteria over time. You can also combine the results from these traditional probabilistic functions with some of the queries detailed above.

- ❖ *Example:* Match FN and LN with Levenshtein values of 0 or 1 and Exact DOB and SSN:

```
select * from (
select distinct a.RecordID RecordID_ME, b.RecordID RecordID_YOU,
a.FN FN_ME, b.FN FN_YOU, Matching.LEVENSHTTEIN(a.FN,b.FN) FN_Value
,a.LN LN_ME, b.LN LN_YOU, Matching.LEVENSHTTEIN(a.LN,b.LN)
LN_Value
,a.DOB DOB_ME, b.DOB DOB_YOU, a.SSN SSN_ME, b.SSN SSN_YOU
```

```

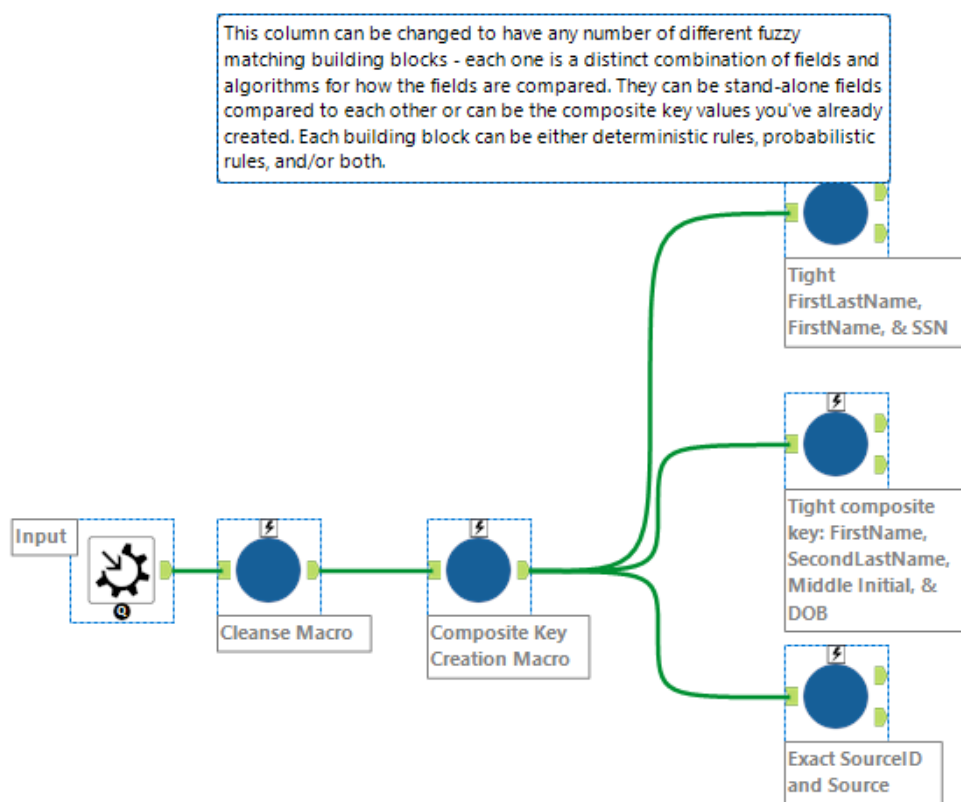
from matching.DataTable a
join matching.DataTable b
on a.RecordID <> b.RecordID and a.DOB = b.DOB and a.SSN = b.SSN
)x
where FN_Value <= 1 AND LN_Value <= 1 and b.RecordID <> a.RecordID
and b.MatchedID <> a.MatchedID

```

- *Match across your data set:* At a certain point, you experience diminishing returns – the time required to manually review each iteration will outweigh any gains made by creating new matching criteria. It's a subjective decision when you've reached this point, but some helpful indicators are if you're not seeing any significant differences with each new iteration or if you're seeing a lot of incorrect matches that would have to be undone when manually reviewing (meaning your criteria has likely become too loose). At this point, it's time to match all rows in your data set!

If matching using a specialized tool (such as RedPoint, Informatica, or Alteryx):

- ▶ **Formatting:** Your first action is to create a column titled RecordID and label each row with its own unique identifier, just like if you were matching using SQL. With a specialized tool, many of the steps described up to this point (standardization, cleansing, composite key creation) may be done in the tool rather than by writing rules in SQL.
- ▶ **Macro creation:** You can create macros in the tool for each of these steps, making them more replicable regardless of source data (such as a cleanse macro with standardized rules to fit all data scenarios and/or a composite key generation macro). Creating a macro for each matching rule lets you plug in the macros that make sense given the data that you're working with. We refer to these matching rule macros as building blocks, which can be used in any combination, and run in parallel, to support the fuzzy matching.
 - ❖ **Example:** If one of your building blocks matches on FirstNameClean, LastNameClean, DOBDigits, and SSNClean, but you don't have SSNs in your data, then you would not want to choose this building block for this matching project.
 - ❖ **Example:** If some of your sources, but not all, have SSNs, you might want a building block that includes SSN as well as other blocks that do not.



- ▶ **Leveraging algorithms:** Part of these tools' benefit is the access they provide to algorithms like Jaro distance or Levenshtein distance, which facilitate matching on more probabilistic patterns. By determining the distance between strings or calculating common characters and the number of transpositions across strings, they can help you identify matches even when your strings are different.
 - ❖ **Example:** Algorithms can detect that the composite key values JOHNSMITH200001016789 and SMITHJOHN200001016789 match, even though one of the strings switches the individual's first and last names.
- ▶ **Using weights and thresholds:** Most specialized tools have built-in weights and thresholds that you can modify, which allows you to prioritize certain fields over others (weights) and helps set the strength of the match (threshold). Having a lower threshold will increase the number of potential matches output, although a lower threshold will also create matches that aren't as strong. This customizability is one of the main advantages of using a specialized tool rather than SQL. The manual that comes with your tool will help you understand how to make these adjustments. When adjusting weights and thresholds, which are unique to each building block, you should iterate with multiple options to see how this alters the results of your match.
 - ❖ **Example:** First, middle, and last names might be given equal weight in one building block, but you may want to emphasize first and last names without discounting middle names entirely in another.
- ▶ **Tracking matches:** Finally, assign the MatchedID from the outputs of the fuzzy matching building blocks to all the rows in newly identified group(s) of the data set. We recommend using the smallest RecordID within the group. This is akin to the process that you'd use if you were working in SQL, although in this case it will be part of your overall workflow and you can leverage your tool to assign the smallest RecordID value to the members in each group – setting the MatchedID as the RecordID when there are no other rows matching to the group.
- ▶ **Match across your data set:** There will be a point where you will have enough building blocks that adding new ones won't identify enough new matches to justify their inclusion – and might even introduce incorrect matches. When you get to this point, it is time to push all the rows of your data set through the workflow!

Match ID	Record ID
0001	0001
0002	0002
0004	0004
0004	0006
0004	0009
0011	0011
0012	0012
0012	0013
0014	0014
0014	0019

Step 6: Manually Review

After completing your first round of fuzzy matching, you review your results manually. This provides you with insight into how your model performed and the extent to which there was overmatching (unique records matched as the same person) and undermatching (multiple

Manual review should always begin with reviewing undermatched files. Once you've completed the initial manual review of your fuzzy matching results, you'll apply the changes back to the matched table and then look for overmatches.

records for the same person went unmatched). No matter how good your fuzzy matching solution is, it will miss linking records that should have been linked or erroneously link records that should not have been linked – attempting to find these during your manual review process and applying corrective measures will help strengthen your confidence in your results. Iterate in your attempts to find undermatches and overmatches – don't assume you can find them all with a single query. This step is crucial and should never be skipped; it's especially important to manually review the performance of your use cases at the start of a project or when you're developing your matching building blocks.

Manual review should always begin with reviewing undermatched files. Once you've completed the initial manual review of your fuzzy matching results, you'll apply the changes back to the matched table and then look for overmatches.

- ▶ **Undermatching:** Undermatches can be difficult to determine since you don't necessarily know what didn't match if it's not matched (you don't know what you don't know) but creating groups of rows can help you see patterns. A good place to start is with the SourceID; make sure all rows with the same SourceID have the same MatchedID to ensure that you're not undermatching. If you have access to an algorithm like Soundex, you can also use the difference function to try to find undermatches. Finally, you can identify undermatches by applying a looser criterion to your results than you originally used when fuzzy matching.
 - ❖ **Example:** Imagine you have two rows, one of which is SMITH, JOHNM 2000-05-02 123456789 and the other is SMITH, JOHN 2000-05-12 123456798. If your original matching criteria included exact first name, last name, DOB, and SSN, these rows would not be assigned the same MatchedID. But by applying looser rules that match on exact last name, exact 3 characters of the first name, similar DOB (like month and year, but not day), and similar SSN (rather than exact DOB and exact SSN), you might find rows that weren't matched due to data entry errors originally but should have been.
- ▶ **Overmatching:** Reviewing groups of rows that were matched together but have multiple distinct values, like DOB and first name, can help you make sure that the MatchedID associated with those rows is accurate and you haven't overmatched unique individuals to a single MatchedID. Overmatches are more common when the criteria used to match

is less restrictive or when twins are present in your data (this is particularly true if your data set includes minors).

- ❖ **Example:** Create and run a SQL query to look for linked groups that have two or more distinct DOB values but the same first and last name combination. Since this is within the same linked group, it can help you make sure you haven't overmatched.

There's no set rule about how many rows you should manually review when fuzzy matching. To a certain extent, this will depend on the size of the data sets you're working with. If there are fewer than 1,000 total rows, you might want to check all rows. With bigger data sets, iterative manual review tables may be generated, and you have to decide which ones make the most sense to review based on the rules you established. Your organization might have its own rules regarding the manual review process, so make sure to check if guidance exists. This might specify the number of rows to review or provide guidance about when to force matches.

While you can use whatever tool you see fit for this step, we've found Excel (or another spreadsheet-based program) to be ideal when manually reviewing. Conditional formatting functions let you color code each matched group of rows to differentiate them from the next group. Custom sorting features can be useful to search for, order, and view specific values. Below we detail how this type of program can be used to identify and correct undermatches and overmatches:

- **Undermatching Manual Review Table:** If you are looking at groups that should have matched but didn't (undermatches), you won't see the same MatchedID values for the rows that you think should be grouped together. You can use conditional formatting to color your rows to show the group that you want to evaluate; alternating colors between different linked rows allows for a quick visual understanding of which rows below in which group. In this example, the smallest RecordID value within the group has been highlighted and should match the values in column B.

Conditional formatting functions let you color code each matched group of rows to differentiate them from the next group.

	A	B	C	D	E	F	G	H	I	J	K	L	M
	Update To MasterID	MatchedID	LN	FN	MN	Sex	SSN	DOB	DataSrc	SourceID	RecordID	Group#	Conditional Formatting Flag
1													
2	434196	782173	SMITH	ANNA		F		5/15/2005	ES	237690	782173	1	1
3	434196	782173	SMITH	ANNA		F	123456789	5/15/2005	HCA	1773056	972262	1	1
4		434196	SMITH	ANNABELLE		F	123456789	5/15/2005	HA	54368	434196	1	1
5		782000	SMITH	TOM		M		1/15/2001	HA	237000	782000	2	0
6	782000	972000	SMITH	THOMAS		M	987654321	1/15/2001	HCA	1773000	972000	2	0

- **Correcting Undermatches:** You can force the match of the two groups by assigning the MatchedID value in Column A. In this case, we have opted to take the smaller of the MatchedID values to assign to the entire group.

	A	B	C	D	E	F	G	H	I	J	K	L	M
	Update To MasterID	MatchedID	LN	FN	MN	Sex	SSN	DOB	DataSrc	SourceID	RecordID	Group#	Conditional Formatting Flag
1													
2	434196	782173	SMITH	ANNA		F		5/15/2005	ES	237690	782173	1	1
3	434196	782173	SMITH	ANNA		F	123456789	5/15/2005	HCA	1773056	972262	1	1
4		434196	SMITH	ANNABELLE		F	123456789	5/15/2005	HA	54368	434196	1	1
5		782000	SMITH	TOM		M		1/15/2001	HA	237000	782000	2	0
6	782000	972000	SMITH	THOMAS		M	987654321	1/15/2001	HCA	1773000	972000	2	0
7													

- **Overmatching Manual Review Table:** In overmatch review files, each group of rows determined to be linked shares the same MatchedID (column B) and the same color (blue or green). Typically, the value of the MatchedID should be the smallest of the possible RecordIDs (column K) within the group. In this example, the smallest RecordID value within the group has been highlighted in column K and should match the values in column B.

	A	B	C	D	E	F	G	H	I	J	K	L	M
	Update To MasterID	MatchedID	LN	FN	MN	Sex	SSN	DOB	DataSrc	SourceID	RecordID	Group#	Conditional Formatting Flag
1													
2		1115165	DOE	JOHN	J	M	123456789	1/1/2006	HA	37771	1115165	1	1
3		1115165	DOE	JOHN	J	M	123456789	1/1/2005	HA	37760	1122718	1	1
4		1115165	DOE	JOHN	J	M	123456789	1/1/2005	HCA	1255443	1542722	1	1
5		1115165	DOE	JOHN	J	M	123456789	1/1/2005	HCA	1255443	1542723	1	1
6		1115165	DOE	JOHN	JAMES	M		1/1/2005	ES	23174	1755524	1	1
7	1545393	1115465	JONES	JANE	M	F	121212121	4/1/1970	HCA	2056214	1545393	2	0
8		1115465	JONES	JANE	M	F	111111111	5/1/1995	HA	42845	1115465	2	0
9		1115465	JONES	JANE	M	F	111111111	5/1/1995	HCA	708038	1496141	2	0
10	1609139	1115465	JONES	JANE	M	F	222222222	3/1/1992	HCA	3664759	1609139	2	0
11	1782741	1115465	JONES	JANE	MARIE				SO	140043	1782741	2	0
12	1793504	1115465	JONES	JANE	M				SO	214205	1793504	2	0
13	1793504	1115465	JONES	JANE	MARIE				SO	214205	1793505	2	0
14		1116267	JOHNSON	BOBBIE	D	F	555555555	7/1/1999	HA	55462	1116267	3	1
15		1116267	JOHNSON	BOBBIE	DANIELLE	F		7/1/1999	VS	145613	1459037	3	1
16	1506877	1116267	JOHNSON	BOBBY	D	M	232323232	1/1/1953	HCA	699982	1506877	3	1
17		1116267	JOHNSON	BOBBIE	D	F	555555555	7/1/1999	HCA	700060	1506881	3	1
18	1640919	1116267	JOHNSON	BOBBY	D	M	565656565	2/1/1968	HCA	4388870	1640919	3	1
19	1773631	1116267	JOHNSON	BOBBY	DIONTAE				SO	173941	1773631	3	1

- **Correcting Overmatches:** If you find an overmatch, remove that row from the group by setting the individual's MatchedID to their own RecordID (as shown in column A below). Within the green group, you can see that only two of the originally matched rows will be kept as part of the overall MatchedID group. The other rows were overmatched and will be split out into other groups by repopulating the MatchedID value that they should be set to (column A).

	A	B	C	D	E	F	G	H	I	J	K	L	M
	Update To MasterID	MatchedID	LN	FN	MN	Sex	SSN	DOB	DataSrc	SourceID	RecordID	Group#	Conditional Formatting Flag
1		1115165	DOE	JOHN	J	M	123456789	1/1/2006	HA	37771	1115165	1	1
2		1115165	DOE	JOHN	J	M	123456789	1/1/2005	HA	37760	1122718	1	1
3		1115165	DOE	JOHN	J	M	123456789	1/1/2005	HCA	1255443	1542722	1	1
4		1115165	DOE	JOHN	J	M	123456789	1/1/2005	HCA	1255443	1542723	1	1
5		1115165	DOE	JOHN	JAMES	M		1/1/2005	ES	23174	1755524	1	1
6	1545393	1115465	JONES	JANE	M	F	121212121	4/1/1970	HCA	2056214	1545393	2	0
7		1115465	JONES	JANE	M	F	111111111	5/1/1995	HA	42845	1115465	2	0
8		1115465	JONES	JANE	M	F	111111111	5/1/1995	HCA	708038	1496141	2	0
9	1609139	1115465	JONES	JANE	M	F	222222222	3/1/1992	HCA	3664759	1609139	2	0
10	1782741	1115465	JONES	JANE	MARIE				SO	140043	1782741	2	0
11	1793504	1115465	JONES	JANE	M				SO	214205	1793504	2	0
12	1793504	1115465	JONES	JANE	MARIE				SO	214205	1793505	2	0
13	1773631	1116267	JOHNSON	BOBBY	DIONTAE				SO	173941	1773631	3	1
14	1640919	1116267	JOHNSON	BOBBY	D	M	565656565	2/1/1968	HCA	4388870	1640919	3	1
15		1116267	JOHNSON	BOBBIE	D	F	555555555	7/1/1999	HCA	700060	1506881	3	1
16	1506877	1116267	JOHNSON	BOBBY	D	M	232323232	1/1/1953	HCA	699982	1506877	3	1
17		1116267	JOHNSON	BOBBIE	DANIELLE	F		7/1/1999	VS	145613	1459037	3	1
18		1116267	JOHNSON	BOBBIE	D	F	555555555	7/1/1999	HA	55462	1116267	3	1
19		1116267	JOHNSON	BOBBIE	D	F	555555555	7/1/1999	HA	55462	1116267	3	1

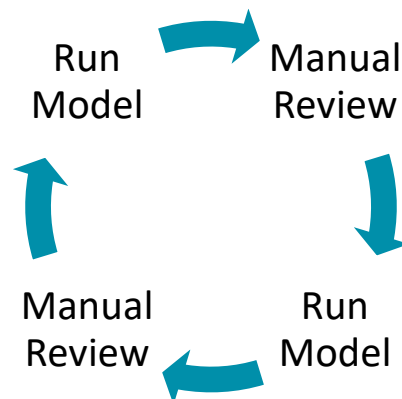
- *Correcting Overmatches Continued:* If you find that your original group has individuals with multiple rows that should be matched, make sure that the new group you create has the same MatchedID (column A). Sometimes when you break groups apart that were overmatched, you need to keep rows together as a new MatchedID group. Below, the bottom two green rows are assigned the same value in column A, which means they will be considered a linked group moving forward.

	A	B	C	D	E	F	G	H	I	J	K	L	M
	Update To MasterID	MatchedID	LN	FN	MN	Sex	SSN	DOB	DataSrc	SourceID	RecordID	Group#	Conditional Formatting Flag
1		1115165	DOE	JOHN	J	M	123456789	1/1/2006	HA	37771	1115165	1	1
2		1115165	DOE	JOHN	J	M	123456789	1/1/2005	HA	37760	1122718	1	1
3		1115165	DOE	JOHN	J	M	123456789	1/1/2005	HCA	1255443	1542722	1	1
4		1115165	DOE	JOHN	J	M	123456789	1/1/2005	HCA	1255443	1542723	1	1
5		1115165	DOE	JOHN	JAMES	M		1/1/2005	ES	23174	1755524	1	1
6	1545393	1115465	JONES	JANE	M	F	121212121	4/1/1970	HCA	2056214	1545393	2	0
7		1115465	JONES	JANE	M	F	111111111	5/1/1995	HA	42845	1115465	2	0
8		1115465	JONES	JANE	M	F	111111111	5/1/1995	HCA	708038	1496141	2	0
9	1609139	1115465	JONES	JANE	M	F	222222222	3/1/1992	HCA	3664759	1609139	2	0
10	1782741	1115465	JONES	JANE	MARIE				SO	140043	1782741	2	0
11	1793504	1115465	JONES	JANE	M				SO	214205	1793504	2	0
12	1793504	1115465	JONES	JANE	MARIE				SO	214205	1793505	2	0
13		1116267	JOHNSON	BOBBIE	D	F	555555555	7/1/1999	HA	55462	1116267	3	1
14		1116267	JOHNSON	BOBBIE	DANIELLE	F		7/1/1999	VS	145613	1459037	3	1
15	1506877	1116267	JOHNSON	BOBBY	D	M	232323232	1/1/1953	HCA	699982	1506877	3	1
16		1116267	JOHNSON	BOBBIE	D	F	555555555	7/1/1999	HCA	700060	1506881	3	1
17	1640919	1116267	JOHNSON	BOBBY	D	M	565656565	2/1/1968	HCA	4388870	1640919	3	1
18	1773631	1116267	JOHNSON	BOBBY	DIONTAE				SO	173941	1773631	3	1
19		1116267	JOHNSON	BOBBY	DIONTAE				SO	173941	1773631	3	1

Step 7: Iterate

Using what you learned from the manual review process in Step 6, iterate on your original matching criteria to minimize the number of undermatches and overmatches in your data set. Create and run different matching rules and use different use cases until you feel comfortable running the model on your full data set. Then go back and compare your pre-match and post-match files and see how your model performed. You'll likely do this multiple times until you feel comfortable with your model. Even the best model will result in over and under matches. Doing manual review checks will likely result in corrections to the MatchedID values pushed by your model. Be sure to dedicate plenty of time to perform manual review checks after the full data set is matched.

While 'iterate' is listed as the final step in our conceptual framework, doing this will take you back to the fuzzy matching and manual review steps (steps 5 and 6). Furthermore, this iterative process is described in those steps (e.g., "expand your matching criteria" in step 5); we simply use this step to call attention to the iterative nature of this process.



At this stage, you have the knowledge base to fuzzy match – now it's just a matter of patience, curiosity, and enjoying the challenge that comes with iterating. Congratulations!

Case Study: Lessons Learned on a Fuzzy Matching Project

One project where we used fuzzy matching involved working with data from a housing authority, a state Medicaid system, two school districts, and a sheriff's office. On this project, our aim was to better understand the extent to which people who engage with one system engage with other systems and how outcomes in one system can create ripple effects for individuals across systems. In this section, we outline a few of the challenges we faced when matching administrative data and the solutions we developed to show what fuzzy matching can look like in practice.

On this project, we quickly discovered that the housing authority data had the same person appearing in their demographic data with different SourceIDs. This was a byproduct of the program's design, where a head of household could put their name on multiple housing waitlists, which resulted in a unique ID associated with each waitlist. Not having single SourceIDs for housing authority data made matching those data to all other data sets unreliable and time consuming (i.e., a single row of Medicaid data would match to all rows of housing data and result in multiple ID values for a single person). Our solution was to fuzzy match the housing authority data as its own data set before matching it to other data sources. For this model, we opted to match with three deterministic building blocks.

Exact LastName,
FirstName, DOB

Exact DOB, SSN,
AddressNumbers

Exact SSN,
LastName,
FirstName, DOB

The result of the match was a single MatchedID we set as the SourceID for rows associated with the housing authority, which in turn allowed us to better match those data to other data sets.

Another challenge that we faced was that booking data from the sheriff's office only provided the first and last name of each person. Since we had so few fields to rely on from this data source, we decided to use the booking data as part of a second round of matching to avoid overmatching on these limited fields. The first round used all other data sources to create a MatchedID field, which we used as the SourceID in the second round of matching. The model we used in this first round of matching included five building blocks.

First LastName, FN,
SSN

First LastName,
First FirstName,
DOB

Exact Source and
SourceID

Exact DOB and LN
+ Tight FirstName
with Common
Nicknames

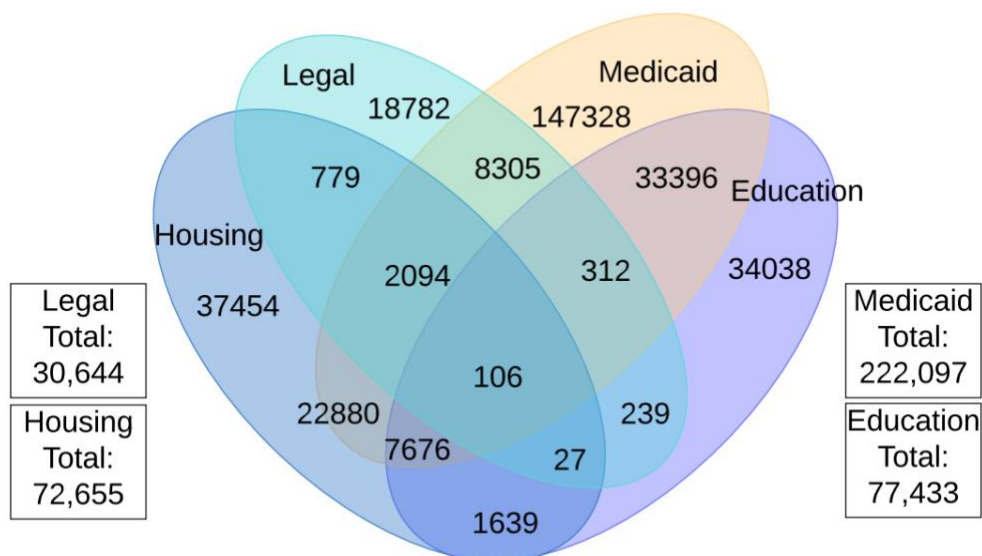
Exact SSN and
FirstName3

In the second round, we compared the SourceID from the first round of matching to the booking data SourceID. To do this, we chose to limit our rules to only allow matching to a different data set in the second round, rather than allow matching within a data set, to ensure that the matching performed in round one was not undone in the second iteration. We also limited overmatching by restricting matches to individuals who were 18 or older during the study window according to the data used in the first round of matching. Since children are unlikely to appear in sheriff's office booking records, we excluded them from the second round of matching to lower the chances that a match would be made on name only for two unique individuals. Finally, we limited the second round of matching to the strictest possible criteria – an exact match on two separate building blocks.



Both building blocks were used because not all data sources had middle names; if we had used only an exact match on full first, middle, and last names, we would have missed the chance to match on rows that didn't have middle names.

While the challenges we faced were unique to these data sets and the goals of this project, all fuzzy matching projects have their own distinct obstacles. These project-specific obstacles are sometimes obvious and sometimes only visible once you've started working in the data, and you'll develop the most appropriate solutions based on the problem you're trying to solve and the question you're trying to answer. In this case study, the data from the sheriff's office were uniquely challenging and mitigating the risk of overmatching became the highest priority. Here are the overlaps we found in the fuzzy matching of that project.



Frequently Asked Questions

While we hope the framework outlined in this document so far gives you a better sense of the fuzzy matching process and its benefits, you might have some remaining questions. Some of the most common questions about fuzzy matching are answered below – we’ve divided them into questions related to project planning, working with data, and equity and bias considerations. If you have questions that aren’t covered here, please reach out to us at CORE!

Project Planning

Are there times when fuzzy matching simply isn’t the best approach?

- ▶ In some cases, deterministic matching will be the only option based on project needs.
 - ❖ *Example:* If an agency plans to end services for an individual based on obtaining a death certificate with the same name as someone who receives services from their agency, they will want to be 100% sure that the deceased individual and the individual for whom they plan to end services are one and the same.

Which fuzzy matching steps can be automated?

- ▶ This will depend to some extent on the tool that you use. If you’re working in SQL, writing cleanse rules, and creating composite keys allows you to automate these processes in your future work. If you’re using a specialized tool with fuzzy matching capabilities, you can set up modularized building blocks within the tool to plug in as needed to create a sense of automation in the future.

Which fuzzy matching steps can be skipped?

- ▶ Ideally, none of these steps are skipped, but the most disposable step is creating composite keys (step 4), since it’s dependent on your access to algorithms like Jaro distance or Levenshtein distance.
- ▶ Additionally, data profiling (Step 1) is extremely helpful as it allows you to find instances where ‘dirty data’ are routinely entered and remove them before you match, but it’s not critical. Its importance will ultimately depend on the cleanliness of your data.
- ▶ Depending on your needs and the data that you’re working with, you might do a more abbreviated manual review (Step 6). This is especially true if you’re working with a very large data set with millions of records. In this case, you might feel that the number of overmatches and undermatches will more or less equal each other out statistically or that the time required to do a complete manual review is simply not feasible. You’ll know your data, your goals, and the relative tradeoffs best.
- ▶ Depending on your data set and your tool, you might also skip some of the iterative work (Step 7). If you have a very large number of rows to match or are doing the matching in SQL, you might avoid applying progressively looser matches due to time or resource constraints.

How long does each fuzzy matching step take?

- ▶ Any time that you work with data, you spend the most time the first time you do the work. Regardless of whether you use SQL or a specialized tool, the process will go faster in the future once you create a process and write building blocks, assuming you do this in a systematic and replicable way. Doubling the number of rows that you're working with won't necessarily double the amount of time required, you'll just experience a longer run time.
- ▶ Similarly, initial manual reviews take more time compared to later reviews. Over time, you become more familiar with the data and figure out a review process that works best for you. Furthermore, you benefit from the initial files now including historical records, so you likely only need to compare records that are new relative to the last time you performed the matching process.

What specific expertise is needed to do this work?

- ▶ We believe that temperament, rather than specific technical expertise, is the most important element of a successful matching project. Someone who enjoys a challenge and the iterative process will likely be more successful doing this work than someone with technical skills but who doesn't have the patience to spend time repeating the steps associated with fuzzy matching. Familiarity with the tool that you're using and a patient attitude should be enough for a successful project.

How many people need to be assigned to work on a fuzzy matching project?

- ▶ Due to the personally identifiable information contained in most data sets, legal agreements typically require that only one or two people be allowed to see the data. If your agreement allows for it, assigning a second person to the project is a great idea as it allows for reliability checks, which are especially important during the manual review process.

Working in the Data

Are there any unique considerations if minors are present in the data sets that I'm matching?

- ▶ If there are children in your data, you want to be very careful of overmatching twins. Twins can have very similar first names and SSNs, and they tend to have the same last names, DOBs, and addresses. You should be especially vigilant for these cases during the manual review process. While you might come across adult twins your data sets, it is less likely that they'll live at the same address.

You want to be clear about the purpose of the data you output from your fuzzy matching model, which will help you make decisions about how to best proceed

Can I fuzzy match if I only have first and last names in my data?

- ▶ First and last names are theoretically the minimum information required to match administrative data. That being said, the more fields that are available to match on, the more confident you can be in any given match.
- ▶ The nature of the data that you're working with will also change your confidence if first and last names are sufficient. If you're working with statewide data, your confidence in your matches will be lower than if you were working with city-wide or county-wide data. Common first or last names might also reduce your confidence in your matches.

Are there any unique considerations if I'm using address data when matching?

- ▶ Address data follows the same rules outlined in this document, although they can be even dirtier than person data. Address data might be all entered as a single field or parsed into separate fields, and you might have to keep or drop apartment or suite numbers from the addresses. There are tools that can help you standardize and cleanse address data, but they can be cost prohibitive or present privacy concerns if they are cloud-based. Like any other field, you want to be clear about the purpose of the data you output from your fuzzy matching model, which will help you make decisions about how to best proceed (e.g., do you need to geocode the addresses? Do you need address data to identify specific clinics or businesses? How necessary are the address fields to the rest of the fuzzy matching?).

If my data contains lots of different fields, should I match on all of them or just choose the field that is most likely to be unique?

- ▶ The more shared elements that you can match on with tighter criteria, the more confident you can feel in your results. In theory, matching using SSNs can lead to a high degree of confidence, but data entry errors mean that SSNs alone don't guarantee that your matches correspond to the same individual. Combining SSN with DOB, first name, middle name, and last name and using tight matching criteria will result in matches you

can be highly confident in, but you risk missing a lot of potential matches. Loosening your criteria will expand your matches, but it might reduce your confidence in the results. There are tradeoffs to both!

How should I measure my confidence in my matches?

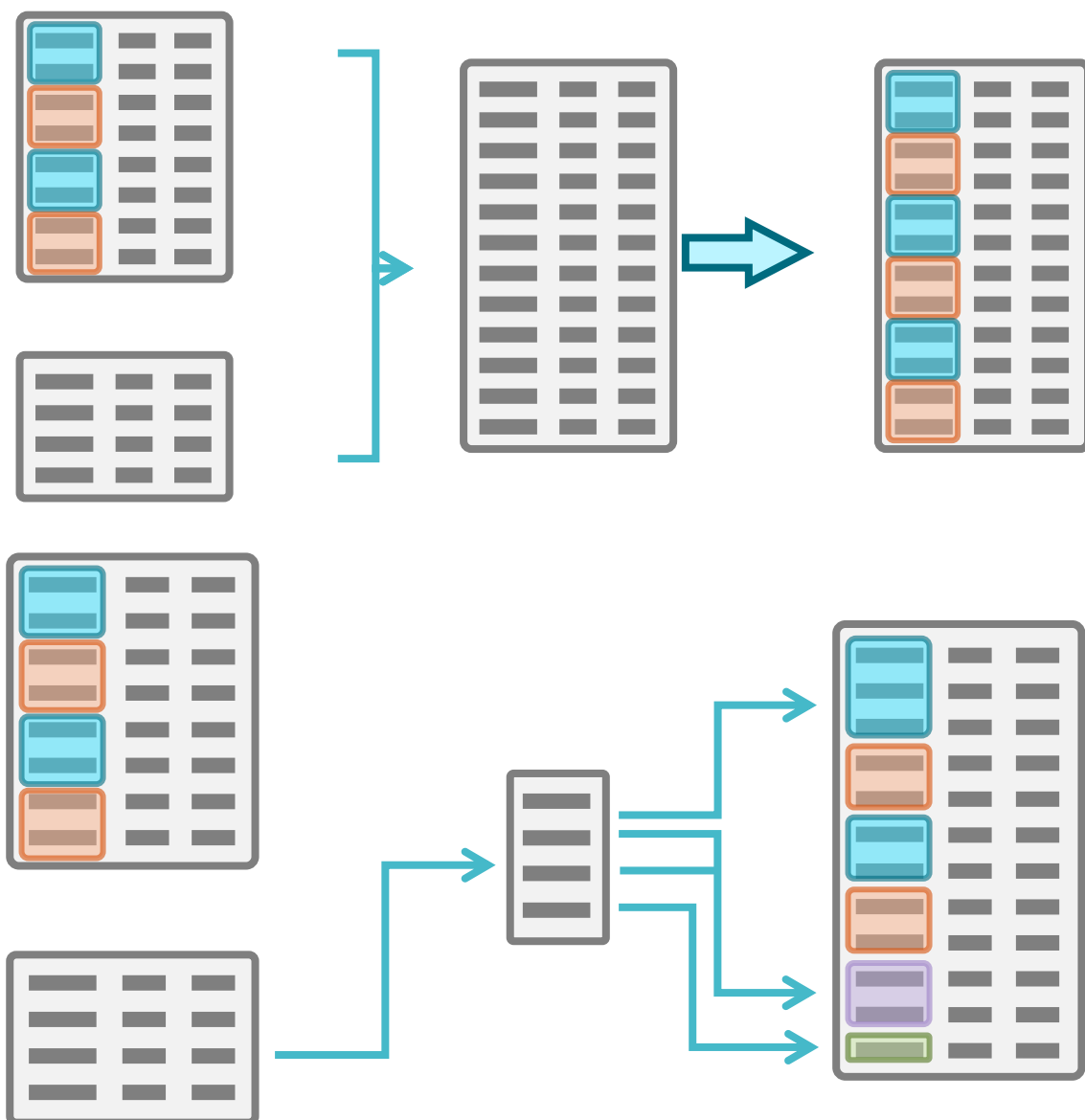
- ▶ If using a specialized fuzzy matching tool, your confidence might be tied to the threshold values that you established for the algorithms that you're using. Beyond these values, there's an innate sense of confidence that you develop when matching within your specific data sets. This confidence isn't something that you can necessarily quantify, but it's no less meaningful than threshold values for gauging the quality of your matching process.
- ▶ Depending on the data that you're working with, there might be known benchmarks that you can use to further verify your matching results.
 - ❖ *Example:* If you're matching Medicaid and Housing Authority data and there's an understanding that 75% of all people assisted by the Housing Authority are enrolled in Medicaid, you'd expect to get a match rate that's close to 75%. If you only got 50%, you'd want to dig into this further on both the program side (confirming the accuracy of the 75% statistic) and the matching side (revisiting your criteria and adjusting to get higher matching rates, if possible).

If I went through the fuzzy matching process once, am I done or will I need to repeat this process in the future?

- ▶ The answer to this question will be project dependent. If your data will not be refreshed, you likely only need to perform the matching process one time. If you're working on a long-term project with periodic data refreshes, you should expect to repeat the matching process after each data refresh.

If I am refreshing the fuzzy match, should I keep my old MatchedID results?

- The answer to this question is also project dependent. Typically, it would make sense to keep your old results if you've already reported on the data matched historically. This way it's easier to match the new rows of data (those received during your data refresh) since you can reference the work you did previously during the manual review process. As the visual below shows (top half), you don't have to keep the old results; they can be discarded, and all the rows can be combined and candidates for a MatchedID value following your data refresh. If you do decide to keep your previous fuzzy matching results, you should flag the historical rows to make sure they don't get a new MatchedID value when you send all your rows through the fuzzy matching process. As seen in the visual below (bottom half), your new rows will either slot into groups that already existed or be part of their own groups if they didn't match to any historical rows.



Equity and Bias Considerations in Fuzzy Matching

How can white supremacy and white dominant culture show up in fuzzy matching?

- ▶ An understanding of the cultural norms and historical contexts associated with the communities represented in the data sets that you're working with is incredibly important. Without this understanding, you can marginalize communities and their members based on naming conventions that are simply unfamiliar to you. Below we highlight just a few examples of considerations that we've found important to keep in mind, although these considerations should be as varied and diverse as the data sets and communities themselves.
 - ❖ *Example:* Some communities adopt two surnames. Both surnames do not always appear in individual's records and the order of the surnames is not always recorded consistently. As a result, you might see the same person with three or more different last names. Similarly, there are often instances where individuals have two first names or have articles as parts of their name ("de," "del," "de la"). The presence of multiple names is one reason that it can be important to separate names into distinct fields and match on different name fields independently, which ensures that name order does not impact your matching. While splitting names into component parts is helpful during the matching process, it's important to ensure that the names are represented as they're written in administrative records in any tables used after matching.
 - ❖ *Example:* Some individuals come from countries where a different alphabet is used, and thus the Roman/Latin alphabet is less common. The anglicization of their names can result in inconsistent spellings across different documents and data sources. Variations in spellings of the same individual's names might result in undermatching and is important to keep an eye out for when manually reviewing.
 - ❖ *Example:* Some communities use a different type of calendar (the Islamic or Hijri calendar is just one example) and are thus less familiar with the Gregorian calendar. This can result in individuals providing the first day of the month or year as their DOB or the DOB for their children in official documents, which then appear in data sets. It is possible that records will overmatch for individuals whose DOB is on the first day of the month or year and is important to consider when manually reviewing.



CORE

Center for Outcomes
Research and Education

Want to Learn More?

If you'd like to learn more about fuzzy matching, CORE's work in general, or are interested in collaborating on a project, please contact Taylor Doren (Taylor.Doren@providence.org).

Appendix A

Additional Suggestions When Cleansing Your Data

The following list elaborates on how you might want to cleanse your data, providing additional examples to those named in Step 3. These suggestions are approaches that we've found helpful when working across data sources, but their relevance will depend on the type of data that you're working with.

Names

Parse common last name suffixes into their own column

- ▶ *Example:* SMITH JR
- ▶ *Cleansed:* CleanLastName: SMITH, LastNameSuffix: JR

Parse names into three columns

- ▶ *Example:* JOHN JACOB SMITH
- ▶ *Cleansed:* FirstName: JOHN, MiddleName: JACOB, LastName: SMITH

Parse the middle initial from the first name column, if there is a single letter after the first name value

- ▶ *Example:* JOHN M
- ▶ *Cleansed:* CleanFirstName: JOHN, CleanMiddleName: M

Create a field with the first four characters from the individual's last name

- ▶ *Example:* JONES-SMITH
- ▶ *Cleansed:* LastName4: JONE
- ▶ *Rationale:* This can be helpful to address different variations of name spellings

Create a field with the first three characters from the individual's first name

- ▶ *Example:* JOHN M
- ▶ *Cleansed:* FirstName3: JOH
- ▶ *Rationale:* This can be helpful to address different variations of name spellings or nicknames

Addresses

Separate address numbers from the rest of an address

- ▶ *Example:* 123 POPLAR STREET
- ▶ *Cleansed:* AddrNumbClean: 123

Make PO Boxes null

- ▶ *Example:* PO BOX 123 as an address
- ▶ *Cleansed:* _____ as an address
- ▶ *Rationale:* Most PO Boxes are considered less reliable data elements, and they cannot be used when geocoding addresses

Remove the hyphen and last 4 digits from the zip code, if present

- ▶ *Example:* 97212-1234
- ▶ *Cleansed:* ZipClean: 97212
- ▶ *Rationale:* The hyphen and extra four digits in zip code data are generally not populated with enough frequency or consistency for them to add additional precision

Dates of Birth

Parse Date of Birth into 3 fields

- ▶ *Example:* 1/1/2000
- ▶ *Cleansed:* YearDOB: 2000, MonthDOB: 1, DayDOB: 1, DOBDigits: 20000101
- ▶ *Rationale:* These fields can be helpful to parse apart and combine into a single standardized field when looking for DOBs within 30 days of each other or with the same days but different years of birth

Social Security Numbers

Remove punctuation from SSNs

- ▶ *Example:* 123-45-67896
- ▶ *Cleansed:* SSNClean: 123456789

If files are inconsistent in listing all 9 digits or only the last 4 of SSNs, create a column for just the last 4 digits

- ▶ *Example:* 123-45-67896
- ▶ *Cleansed:* SSN4: 6789

Appendix B

An Example Workflow: What This Process Looks Like at CORE

This guide was intentionally written to be applicable to organizations with a wide range of available resources and working on projects with different goals. In this section, we share what the process looks like at our organization and the rationale behind some of the decisions we made. This is not intended to be directive; you know your organization and your needs best! Rather, we hope this section provides concrete examples of the workflow and necessary decisions associated with fuzzy matching projects.

What tool do we use?

- ▶ We use Alteryx. We use the built-in tools to build workflows that perform data standardization, data cleansing, and composite key creation.

Why did we choose this tool?

- ▶ We analyzed a range of different tools based on built-in functionality for fuzzy matching and for their additional features. While Alteryx has robust fuzzy matching capabilities, we also chose it based on its spatial components (i.e., addresses) and its ability to work with many different inputs and outputs.

What algorithms do we use?

- ▶ This is completely driven by the project that we're working on and the data that we're using. We base this decision on the data elements available to match on and the cleanliness of the data. When we're ready to fuzzy match, we use an array of matching building blocks that we have already created or create new ones based on project needs.

From start to finish, who all is involved in a fuzzy matching project?

- ▶ A Data Systems Analyst, Project Analyst, and Project Lead are the team members involved in fuzzy matching at CORE. Our process from start to finish is as follows:
 - ❖ First, the Project Lead establishes data sharing agreements. This can take more time than the actual fuzzy matching process!
 - ❖ Once all the agreements are in place, the Project Lead gathers information about the populations being matched and the expected match rate across the data sets being combined.
 - ❖ Our Data Systems Analyst then stages the data to a database, where they also profile the data.

- ❖ If the data are messy and individuals have multiple IDs, the Data Systems Analyst will match a single data set in Alteryx to generate unique IDs for each person.
- ❖ The Data Systems Analyst will then consolidate the data from across data sets and add metadata fields about their sources in a SQL table. The table often gets named something like PreAlteryx. Here is the data definition language (DDL) for an example table:

```
CREATE TABLE [Matching].[PreAlteryx](
    [RecordID] [bigint] NOT NULL,
    [SourceID] [varchar](50) NULL,
    [FN] [varchar](100) NULL,
    [MN] [varchar](100) NULL,
    [LN] [varchar](100) NULL,
    [Sex] [varchar](50) NULL,
    [DOB] [date] NULL,
    [SSN] [varchar](11) NULL,
    [Lang] [varchar](100) NULL,
    [Addr] [varchar](300) NULL,
    [City] [varchar](50) NULL,
    [State] [varchar](50) NULL,
    [Zip] [varchar](20) NULL,
    [DataSrc] [varchar](20) NULL
)
```

- ❖ Next, the Data Systems Analyst will both create new and use existing Alteryx workflows to process the use cases through the cleanse and composite key creation macros, as well as through various fuzzy matching building blocks. They will iterate with different combinations of building blocks until the sequence of building blocks looks good for the use case based on manual review and the model is ready to run on the full data set.
- ❖ A full run of the data set will be pushed through the workflow that has been tested with the matched data set and be output to a SQL table named something like PostAlteryxMatch. Here is the DDL for an example table:

```
CREATE TABLE [Matching].[PostAlteryx](
    [RecordID] [bigint] NULL,
    [SourceID] [varchar](100) NULL,
    [MatchCriteria] [varchar](100) NULL,
    [MatchScore] [smallint] NULL,
    [ID_CORE] [bigint] NULL,
    [DataSrc] [varchar](20) NULL,
    [LN] [varchar](100) NULL,
    [MN] [varchar](100) NULL,
    [FN] [varchar](100) NULL,
    [DOB] [date] NULL,
    [Zip] [varchar](5) NULL,
    [Sex] [varchar](50) NULL,
    [Lang] [varchar](100) NULL,
    [SSN] [varchar](9) NULL,
    [AddrNumbers] INT NULL,
    [City] [varchar](50) NULL,
    [State] [varchar](50) NULL,
    [LN4] [varchar](4) NULL,
    [FN3] [varchar](3) NULL,
    [FirstLN] [varchar](100) NULL,
    [SecondLN] [varchar](100) NULL,
    [Year_DOB] [smallint] NULL,
    [Sector] [varchar](20) NULL
)
```

- ❖ The Data Systems Analyst and a Project Analyst will then do a full manual review of the results, assuming the Project Analyst has permission to view the data.
- ❖ Once the final manual review is complete and agreed upon, the Data Systems Analyst will create a crosswalk of the matched results tied to the SourceID values.
- ❖ The crosswalk will then be joined to the source tables, with the SourceID values dropped and the MatchedID values retained along with the rest of the source data. At this point, the new tables with the attached MatchedID and removed SourceID can be considered de-identified.

CORECenter for Outcomes
Research and Education

The Center for Outcomes, Research, and Education (CORE) is an independent team of scientists, researchers, and data experts with a vision for a healthier, more equitable future. Based in Portland, Oregon, we partner with changemakers and communities to take on today's biggest barriers to better health. Through research, evaluation, and analytics, we provide insights that help shape and sustain healthier systems, policies, and programs.

**5251 NE Glisan St.
Portland, OR 97213
(503) 215-7170**

ProvidenceOregon.org/CORE

Evidence for Change